1983

# A Matching Problem in the Plane

Mikhail J. Atallah
*Purdue University*, mja@cs.purdue.edu

Report Number:
83-462

# A MATCHING PROBLEM IN THE PLANE

Mikhail J. Atallah

Department of Computer Sciences
Purdue University
West Lafayette, Indiana 47907.

**Abstract**

Suppose we are given $2n$ distinct points in the plane, no three of which are collinear, $n$ of which are colored blue and the remaining $n$ are colored red. The problem we consider is that of finding a one to one correspondence between red and blue points such that if we join every pair of corresponding points by a straight line segment, then no two of the resulting $n$ segments intersect. We give an $O(n \log^2 n)$ time algorithm for computing the desired one to one correspondence. The algorithm can be used in the context of motion-planning.

## 1. Introduction

Consider the following problem: Given the initial positions of $n$ identical objects in the plane (represented by $n$ distinct red points), and given $n$ destinations (represented by $n$ distinct blue points), assign to every object a destination, one object per destination, in such a way that if every object moves to its destination on a straight line segment then no two segments intersect (this eliminates the possibility of a collision between two moving objects). In other words, we want to draw $n$ nonintersecting straight line segments, each of which joins a red point to a blue one. We show that this is always possible if no three of the $2n$ points are collinear, and give an $O(n \log^2 n)$ time algorithm for finding the desired $n$ straight line segments.

Throughout the paper, we assume that no three of the given $2n$ points are collinear. For convenience, we assume that $n = 2^k$. All logarithms are to the base 2.

## 2. Preliminaries

In this Section we present a few preliminary observations, and then we review some known results which will be needed later in the paper.

**Lemma 1** There exists a straight line $l$ that divides both red and blue points in half.

*Proof:* For every angle $\alpha$, we define $f(\alpha)$ as follows: Starting at $y = -\infty$, slide (upward) a line parallel to direction $\alpha$ until it first partitions the blue points in half (we are ignoring the situation where the slope of the line is such that it cannot partition the blue points exactly in half, since the proof can easily be modified to handle this case). Let $R(\alpha)$ be the number of red points to the left of the resulting line, $S(\alpha)$ the number of red points to its right. Then

$f'(\alpha)=R(\alpha)-S(\alpha)$. Observe that $f(\alpha)$ takes only even values. Since we are assuming that no three points are collinear, $f(\alpha)$ changes by "jumps" of size two, i.e. $\lim_{\varepsilon\to 0}|f(\alpha+\varepsilon)-f(\alpha)|\leq 2$. If $f(0)=0$ then we're done, otherwise assume without loss of generality that $f(0)>0$. Note that $R(\alpha+\pi)=S(\alpha)$ and $S(\alpha+\pi)=R(\alpha)$, and therefore $f(\pi)=-f(0)<0$. Since $f(0)>0$, $f(\pi)<0$, and $f(\alpha)$ changes by "jumps" of size two and takes only even values, it follows that there exists a value $\alpha'$ between 0 and $\pi$ for which $f(\alpha')=0$. ▪

**Theorem 2** It is possible to draw $n$ nonintersecting straight line segments, each of which joins a red point to a blue one.

*Proof A:* Recursive application of Lemma 1 yields a subdivision of the plane into $n$ convex faces each of which contains one blue point and one red point which can be joined by a straight line (that no intersection will occur is guaranteed by the convexity of each face). ▪

*Proof B:* Let $S$ be a set of $n$ segments each of which joins a red point to a blue one and no two of which have a common endpoint. Let the *cost* of $S$ be the sum of the Euclidean lengths of the $n$ segments in it. If $S$ has minimum cost, then no two segments in $S$ intersect, because if they did then the fact that the sum of the lengths of the two diagonals of a convex quadrilateral is larger than the sum of the lengths of any two opposite sides can be used to further decrease the cost of $S$, a contradiction. ▪

Both *proofs A* and *B* result in polynomial time algorithms for finding the desired $n$ nonintersecting segments. *Proof B* suggests using the known techniques for finding minimum-weight matchings, which result in an $O(n^3)$ time algorithm (see Ch.11 in [PS]). *Proof A* results in an $O(n^2\log n)$ time algorithm, as follows: Finding a line which divides both red and blue points in half can be

done in time $O(n^2 \log n)$ (the details are left to the reader). This implies that, if we let $T(n)$ be the running time of the divide-and-conquer algorithm suggested by proof $A$, then

$$T(n) \leq 2T(n/2) + c_1 n^2 \log n \quad \text{if } n>2, \text{ and}$$
$$T(2) = c_2$$

This implies that $T(n)=O(n^2 \log n)$.

The next Section will present the main result of this paper: An $O(n \log^2 n)$ time algorithm.

We now recall some known results which will be needed in this paper.

Overmars and Van Leeuwen have designed a data structure (they call it an *augmented tree structure*) for storing $n$ points according to (say) their $x$-coordinate, in such a way that the convex hull is stored at the root of the structure and is maintained as points are inserted/deleted at a cost of $O(\log^2 n)$ time per insertion/deletion (see reference [OV] for the details of how this intersesting data structure functions). They also show that, if the points in two distinct augmented tree structures $T_1$ and $T_2$ are separable by a line $V$ parallel to the $y$-axis, then a tangent $PQ$ common to both the convex hull of the points in $T_1$ and to that of the points in $T_2$ (as in Fig. 1) can be found in time $O(\log n)$ (Theorem 3.2 in reference [OV]).

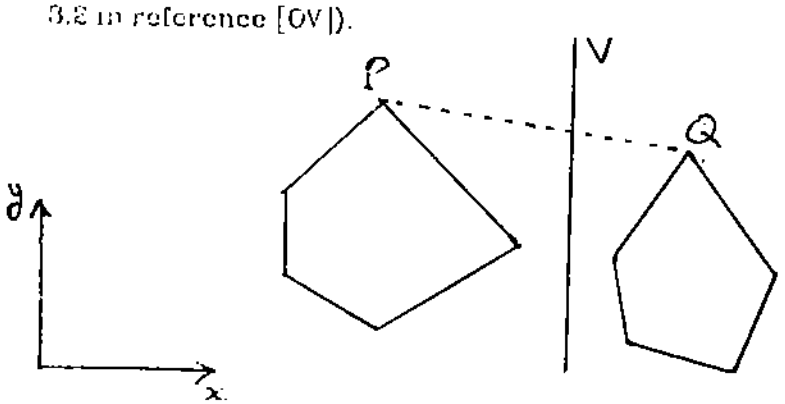


Figure 1

Chazelle and Dobkin [CD] have shown that it is possible to find the intersection of a line and a convex polygon in time $O(\log n)$, and that detecting whether

two convex polygons are disjoint can also be done in time $O(\log n)$.

### 3. The Algorithm

We assume that, initially, the red points are stored in an augmented tree structure (call it $TR$) according to their $x$-coordinate (for convenience, we assume that no two points have the same $x$-coordinate). As already mentioned, structure $TR$ also contains a description of the convex hull (call it $HR$) of the red points, and it supports insertion and deletion operations in time $O(\log^2 n)$. We assume that a similar structure is available for the blue points (we call it $TB$, and we call the convex hull of the blue points $HB$). The algorithm we will describe assumes that $TR$ and $TB$ are available (creating them takes time $O(n \log n)$ [OV], while the algorithm itself takes time $O(n \log^2 n)$).

We now give a recursive description of the algorithm for drawing $n$ nonintersecting straight line segments, each of which joins a red point to a blue one. Our description of the algorithm is informal, and we postpone many crucial implementation details till the analysis which comes after this description of the algorithm:

*Step 1:* Test whether $HR$ and $HB$ are disjoint. If they are not disjoint then proceed to Step 2. Otherwise find a line $L_1$ which separates them, then create new structures $TR$ and $TB$ in which the red and (respectively) blue points are stored according to their projections on an axis $L_2$ perpendicular to $L_1$ (Fig. 2), i.e. $L_1$ and $L_2$ become the new $y$ and $x$ axes, respectively (in the "old" $TR$ and $TB$ structures, the points were stored according to their projections on the old $x$-axis).

Now, repeat the following (i)-(ii) until no more points remain:

(i) Find a line tangent to both $HR$ and $HB$ at, say, points $P$ and $Q$, respectively. It is important that both $HR$ and $HB$ be on the same side of the line, e.g. below it (Fig. 2).

(ii) Join $P$ to $Q$ by a straight-line segment (this is one of the desired $n$ segments). Then delete $P$ from $TR$, $Q$ from $TB$.

return

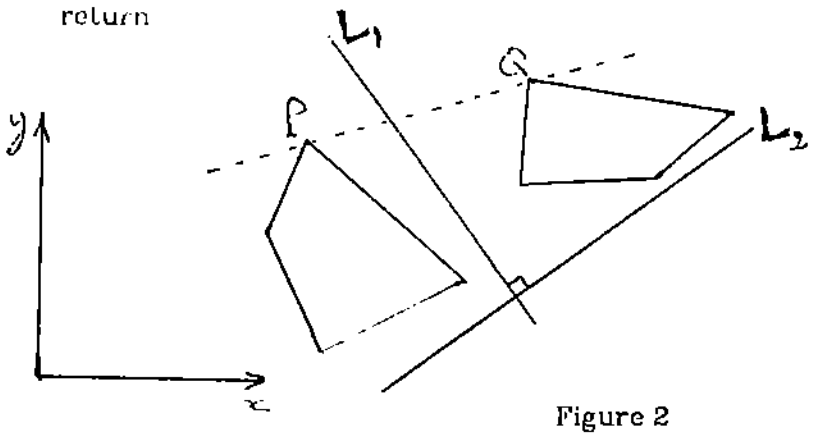


Figure 2

*Step 2:* Let $a$ and $b$ be (respectively) the smallest and largest $x$-coordinates among the red points, and let $v$ and $w$ be (respectively) the smallest and largest $x$-coordinates among the blue points. Repeat the following (i)-(iii) until either no points remain, or one of the two intervals $[a,b]$ and $[v,w]$ contains the other (as in Figure 3,a):

(i) Find a common tangent to $HR$ and $HB$ such that $HR$ and $HB$ are on the same side of this tangent. Let $P$ and $Q$ be (respectively) the red and blue points on this tangent (Fig. 3,b).

(ii) Join $P$ and $Q$ by a straight-line segment (this is one of the desired $n$ segments).

(iii) Delete $P$ from $TR$, $Q$ from $TB$.

Comment: Note that if one of the intervals $[p,q]$ and $[v,w]$ contains the other then there exists a vertical line (call it $L$) with the property that there are points both to its left and to its right, with as many red points as blue ones to its

left, and similarly to its right (Fig. 3,a).

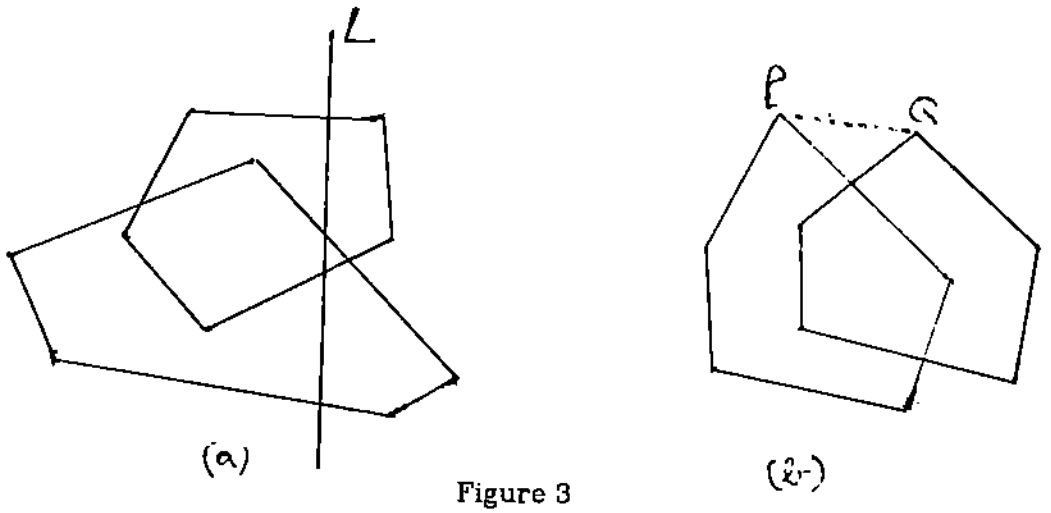


Figure 3

*Step 3:* If no points remain then return, otherwise find vertical line $L$ by performing, in parallel, two "scans" of the remaining points (both red and blue). By "in parallel" we mean that we alternate between one scan and the other, letting them progress simultaneously. One scan starts at the leftmost point and moves rightward, and the other starts at the rightmost point and moves leftward. Both scans "look" for line $L$ until one of them finds it.

Comment: Note that, if $k$ red and $k$ blue points are to the right of $L$, and $k'$ red and $k'$ blue points are to the left of $L$, then the time taken by Step 3 is $O(min(k,k'))$.

*Step 4:* Find $TR_1$, $TR_2$, $TB_1$ and $TB_2$, where $TR_1$ and $TB_1$ are the augmented tree structures of the (respectively) red and blue points to the left of $L$, and $TR_2$ and $TB_2$ are analogously defined for points to the right of $L$. $TR_1$ and $TR_2$ ($TB_1$ and $TB_2$) are obtained by splitting $TR$ ($TB$) about the $x$-component of the vertical line $L$.

*Step 5:* Recursively solve the problem for points to the left of $L$ (using $TR_1$ and $TB_1$), then for points to the right of $L$ (using $TR_2$ and $TB_2$).

(End of Algorithm)

Correctness of the algorithm is obvious. We now show that it runs in time $O(n \log^2 n)$. We do so by showing that the total cost of every one of Steps 1-5 is $O(n \log^2 n)$, where by "total cost" of a Step we mean its cost over all recursive calls.

*Cost of Step 1:* Testing whether $HR$ and $HB$ are disjoint and finding $L_1$ can be done in time $O(\log n)$ [CD], and since this is done $O(n)$ times its total cost is $O(n \log n)$ time. Since a point is involved only once in the "restructuring" of $TR$ and $TB$ the total cost of such restructuring is $O(n \log n)$ time. After this restructuring of $TR$ and $TB$, we can use the $O(\log n)$ time algorithm of Overmars and Van Leeuwen for finding the tangent common to $HR$ and $HB$, and since this is done a total of $O(n)$ times its cumulative cost is $O(n \log n)$. On the other hand, deleting a point from $TR$ or $TB$ takes time $O(\log^2 n)$ and is done $O(n)$ times, and hence the total cost of such deletions is $O(n \log^2 n)$ time. The cumulative cost of Step 1 is therefore $O(n \log^2 n)$ time.

*Cost of Step 2:* If we can show that the common tangent can be found in time $O(\log n)$ then an argument similar to the one given for Step 1 would give a total cost of $O(n \log^2 n)$ time for Step 2. However, we cannot immediately make use of Overmars and Van Leeuwen's already mentioned result for tangent determination, because $HR$ and $HB$ may not be disjoint (Fig. 3,b). Actually, the general problem of finding a common tangent (or reporting that none exists) for two possibly intersecting convex polygons can easily be shown to require time $\Omega(n)$ in the worst case (we leave the details of such a proof to the interested reader). However, here we have the additional information that none of the two intervals $[a,b]$ and $[v,w]$ contains the other. We now use this fact in order to find the desired tangent in time $O(\log n)$. Without loss of generality, assume that $a<v$

(this implies that $w>b$). Let $A$ be the (red) point whose $x$-component is $a$, $W$ be the (blue) point whose $x$-component is $w$. Let $C$ be the blue point such that $AC$ is a supporting line of $HB$ and $HB$ is below it (Fig. 4). Similarly, $D$ is the red point such that $WD$ is a supporting line of $HR$ and $HR$ is below it. Points $C$ and $D$ can be found in time $O(\log n)$ [S]. If line $AC$ ($WD$) intersects $HR$ ($HB$) only at $A$ ($W$) then $AC$ ($WD$) is the desired tangent, so assume that $E$ ($F$) is another point of intersection (Fig. 4). $E$ and $F$ can be found in time $O(\log n)$ [CD]. Now, let $V$ be a vertical line passing through $E$ and let $H_1$ be the portion of $HR$ to the left of $V$. Let $V'$ be a vertical line passing through $F$ and let $H_2$ be the portion of $HB$ to the right of $V'$. The desired tangent is tangent to $H_1$ and $H_2$. Since $H_1$ and $H_2$ are separable by a vertical line (any vertival line between $V$ and $V'$ will do), Overmars and Van Leeuwen's technique can now be used for finding the desired common tangent in time $O(\log n)$.

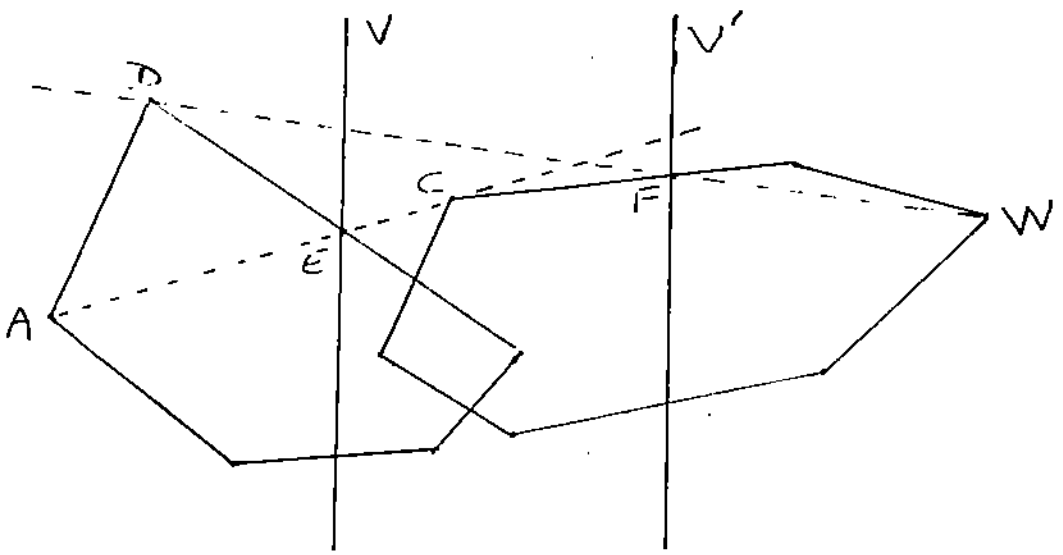


Figure 4

*Cost of Step 3:* Without loss of generality, assume that $k \leq k'$. Charge the $O(k)$ time it takes to find $L$ to the $2k$ points to the right of $L$. Note that whenever a point is charged it ends up in a sub-problem of half the size of the previous sub-

problem it was in. This implies that a point gets charged $O(\log n)$ times, and hence the total cost of Step 3 is $O(n \log n)$ time.

*Cost of Step 4:* Splitting $TR$ ($TB$) into $TR_1$ and $TR_2$ ($TB_1$ and $TB_2$) can be done in time $O(\log^2 n)$ by an easy extension of the techniques of Overmars and Van Leeuwen [OV]. Since there are $O(n)$ such splittings, their total cost is $O(n \log^2 n)$ time.

*Cost of Step 5:* We have already accounted for the costs of the recursive calls.

This completes the proof of the following

**Theorem 3** The above algorithm runs in time $O(n \log^2 n)$.

## 4. Conclusion

Given $2n$ points, no three of which are collinear, $n$ of which are red and the remaining $n$ are blue, we considered the problem of finding $n$ nonintersecting straight line segments, each of which joins a red point to a blue one. We gave an $O(n \log^2 n)$ time algorithm for this problem.

The existence of such nonintersecting segments can no longer be guaranteed if we drop the non-collinearity assumption, as can be seen by considering the case when all red points are on the negative part of the $x$-axis and all blue ones are on the positive part of the $x$-axis.

The following constrained versions of this problem are worth investigating:

(i) There are *obstacles* in the plane, i.e. some areas of the plane are forbidden and no segment can go through them, or

(ii) Points in a given $m$-subset of the red points can only be joined to points in a given $m$-subset of the blue ones ($m < n$).

**References**

[CD] B. Chazelle and D. P. Dobkin, 'Detection Is Easier Than Computation,' *Proc. of 12th Annual ACM Symposium on Theory of Computing*, 1980, pp. 146-153.

[OV] M. H. Overmars and J. Van Leeuwen, 'Maintenance of Configurations in the Plane,' *Journal of Computer and System Sciences*, 1981, pp. 166-204.

[PS] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice Hall, Englewood Cliffs, NJ, 1982.

[S] M. I. Shamos, 'Computational Geometry', Dept. of Computer Sci., Yale U., New Haven, Conn., 1977.